ГЛАВА XB.
Линейные классификаторы

Не трудно придумать новое,
трудно отказаться от старого.
Д.М. Кейнс

Где материя, там геометрия.
И. Кеплер

Линейные решающие модели в задаче бинарной классификации

Рассматриваем задачу с вещественными признаками: $X = \mathbb{R}^n$ и двумя непересекающимися классами: $Y = \{\pm 1\}$ (здесь нам будут удобны такие метки классов), обучающая выборка: $\{(x_i, y_i)\}_{i=1}^m$, хотим использовать линейную модель:

$$a(x) = \text{sgn}(w^{\text{T}} x + b) = \begin{cases} +1, & w^{\text{T}} x + b > 0, \\ -1, & w^{\text{T}} x + b < 0, \end{cases}$$

случай $w^{\text{T}} x + b = 0$ нам тут не особо важен, для определённости можно считать, что в этом случае $a(x) = +1$ (а на обучающем объекте x, что алгоритм выдаёт неверную метку). Заметим, что формально зависимость целевого значения от признаков нелинейная, но модель всё равно называется **линейным классификатором**, и при пополнении признакового пространства фиктивным константным признаком записывается в виде

$$a(x) = \text{sgn}(w^{\text{T}} x). \qquad \text{(XB.01)}$$

Геометрический смысл линейного классификатора показан на рис. XB.1: пространство делится гиперплоскостью на два полупространства (точки одного классификатор считает объектами класса +1, а точки второго – объектами класса –1). Если задачу можно решить линейным классификатором без ошибок, то выборка называется **линейно разделимой**[1].

[1] Это происходит, когда выпуклые оболочки объектов выборки из разных классов не пересекаются.

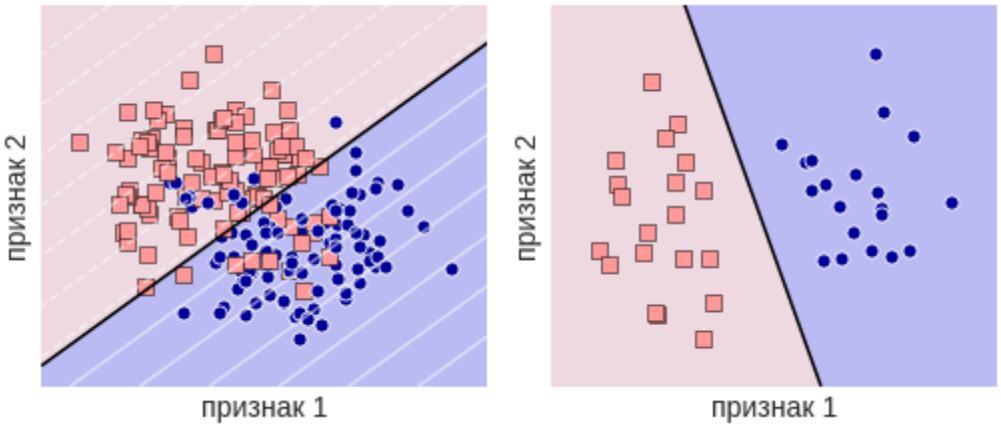


Рис.XB.1. Решающие поверхности линейных классификаторов

При обучении линейного классификатора естественно минимизировать число ошибок, т.е.

$$L(X_{\text{train}}, a) = \sum_{t=1}^{m} L(y_t, a(x_t)) \to \min,$$

где

$$L(y_t, a(x_t)) = \begin{cases} 1, & a(x_t) = y_t \\ 0, & a(x_t) \neq y_t. \end{cases}$$

Такая функция ошибки называется **0-1-loss** и равна числу неверно классифицированных объектов. Однако

- функция не везде дифференцируема по параметрам w модели (XB.01), а там где дифференцируема её производная равна нулю (что делает невозможным использование градиентных методов для минимизации),
- содержит «мало информации»: только число ошибок, а не то, насколько ошибся классификатор (интуитивно, если объект попал в «чужое полупространство», то чем дальше он от разделяющей гиперплоскости, тем больше ошибка),
- её оптимизация здесь – NP-полная задача (поэтому не существует эффективных методов точной оптимизации).

Заметим, что алгоритм (XB.01) совершает ошибку на объекте x_t, если ответ $w^{\mathrm{T}} x_t$ и метка y_t не одного знака, т.е. выполняется неравенство $y_t w^{\mathrm{T}} x_t \leq 0$. Поэтому 0-1-loss переписывается в таком виде:

$$L(y_t, a(x_t)) = \theta(-y_t w^{\mathrm{T}} x_t),$$

где используется индикаторная функция Хэвисайда

$$\theta(z) = I[z \geq 0] = \begin{cases} 1, & z \geq 0, \\ 0 & z < 0. \end{cases}$$

Заметим, что (по логике, описанной выше) чем меньше выражение $z_t = y_t w^{\mathrm{T}} x_t$, тем более грубая ошибка совершается, это выражение называется **зазором на объекте (margin)**. Для устранения описанных недостатков функцию ошибки L аппроксимируют, а чаще оценивают сверху т.н. **суррогатной функцией** L' **(surrogate loss functions)** и подменяют задачу оптимизации:

$$\sum_{t=1}^{m} L(y_t, a(x_t)) \leq \sum_{t=1}^{m} L'(y_t, a(x_t)) \to \min .$$

А чем больше зазор, тем уверенней даётся верный ответ.

Из того, что $L = \theta(-z_t)$, проще искать функцию $f(z)$: $\theta(-z) \leq f(-z)$ и решать суррогатную задачу оптимизации

$$\sum_{t=1}^{m} \theta(-z_t) \leq \sum_{t=1}^{m} f(-z_t) \to \min .$$

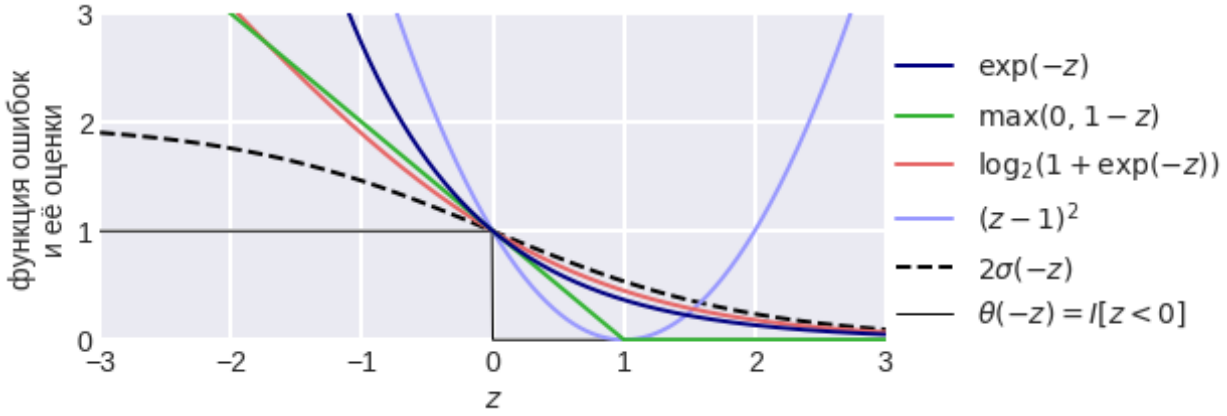


Рис. XB.2. 0-1-loss и суррогатные функции.

На рис. XB.2 показаны примеры суррогатных функций. График функции $\theta(-z)$ представляет из себя ступеньку (тонкая чёрная линия), графики остальных функций лежат не ниже этой ступеньки. Дальше мы увидим, что функция

$f(-z) = \max(0, 1-z)$ используется в методе SVM,

$f(-z) = \exp(-z)$ – в бустинге,

$f(-z) = \log(1+\exp(-z))$ – в логистической регрессии.

Подход суррогатных функций ошибки один из основных в машинном обучении.

Все эти функции гладкие, с ними проще решать задачу оптимизации[1] (естественно, используют также регуляризацию). Таким образом, NP-полную задачу мы подменяем выпуклой задачей оптимизации (что не гарантирует, конечно, точного решения исходной задачи).

Персептронный алгоритм

В качестве первого простого примера рассмотрим персептронный алгоритм, в котором используется суррогатная функция $f(-z) = \max(0, -z)$. Заметим, что она не оценивает сверху 0-1-loss, но довольно логична: её значение равно нулю при верной классификации и возрастает при уменьшении z. Тогда обучение линейного классификатора соответствует решению такой оптимизационной задачи

$$\sum_{i=1}^{m} \max[0, -y_i(w^{\mathrm{T}} x_i)] \to \min. \qquad \text{(XB.02)}$$

Персептроном называется линейный классификатор, веса которого получается в результате применения в (XB.02) метода стохастического градиентного спуска (SGD), который итерационно пересчитывает веса по такой формуле

$$w \leftarrow w + \eta \begin{cases} 0, & \operatorname{sgn}(w^{\mathrm{T}} x_i) = y_i, \\ +x_i & w^{\mathrm{T}} x_i \le 0, y_i = +1, \\ -x_i, & w^{\mathrm{T}} x_i \ge 0, y_i = -1, \end{cases}$$

Можно пояснить, почему персептронный алгоритм «работает». Предположим, что на объекте из класса +1 произошла ошибка (для объекта другого класса рассуждения аналогичные), в этом случае скалярное произведение $w^{\mathrm{T}} x_i \le 0$, а хотелось бы, чтобы оно было положительно, но тогда

$$w^{\mathrm{T}} x_i \le 0, y_i = +1 \Rightarrow w \leftarrow w + \eta x_i \Rightarrow w^{\mathrm{T}} x_i \leftarrow w^{\mathrm{T}} x_i + \eta x_i^{\mathrm{T}} x_i,$$

т.е. после поправки весов мы увеличиваем скалярное произведение на

$$x_i^{\mathrm{T}} x_i = \| x_i \|_2^2 > 0$$

(обратим внимание, что в пополненном признаковом пространстве векторы-

[1] Про суррогатные функции можно прочитать в Bartlett P. L., Jordan M. I., McAuliffe J. D. Convexity, classification, and risk bounds //Journal of the American Statistical Association. – 2006. – Т. 101. – №. 473. – С. 138-156.

описания объектов ненулевые, поэтому и квадрат L2-нормы положительный). Если несколько раз делать шаг SGD только на этом объекте, то скалярное произведение в какой-то момент станет положительным, т.е. объект будет верно классифицирован. Есть теорема Новикова: если две конечные выборки линейно разделимы, то разделяющая поверхность находится персептронным алгоритмом за конечное число шагов[1].

Рассмотрим решение системы линейных неравенств персептронным алгоритмом (нетрудно видеть, что построение безошибочного линейного классификатора и поиск хотя бы одного решения системы линейных неравенств – эквивалентные задачи). Система

$$\begin{cases} w^{\mathrm{T}} x_i > 0, & i \in I_{+1}, \\ w^{\mathrm{T}} x_i < 0, & i \in I_{-1}, \end{cases}$$

легко переписывается в систему «с одним знаком»:

$$\begin{cases} +w^{\mathrm{T}} x_i > 0, & i \in I_{+1}, \\ -w^{\mathrm{T}} x_i > 0, & i \in I_{-1}, \end{cases}$$

и после переобозначений $x_i \leftarrow -x_i$ при $i \in I_{-1}$ получаем систему $w^{\mathrm{T}} x_i > 0$, $i \in I_{+1} \cup I_{-1}$, но для её решения персептронный алгоритм переписывается в совсем простом виде:

$$w \leftarrow w + x_i \text{ при } w^{\mathrm{T}} x_i \le 0,$$

т.е. когда i-е неравенство не выполняется, просто прибавляем x_i к вектору параметров. Рассмотрим, например систему

$$\begin{cases} 2w_1 + w_2 > 0 \\ -w_1 > 0 \\ w_1 - w_2 < 0 \\ -2w_1 - 2w_2 < 0 \end{cases}$$

преобразуем её в вид

$$\begin{bmatrix} 2 & 1 \\ -1 & 0 \\ -1 & 1 \\ 2 & 2 \end{bmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} > 0.$$

[1] Здесь она подробно не разбирается, т.к. не представляет практического интереса.

Если теперь перебирать объекты по циклу, начиная с первого, и стартовать с нулевого вектора параметров, то сначала не будет выполнено первое неравенство, поэтому

$$w \leftarrow (0,0) + (2,1) = (2,1),$$

потом не выполнено второе, поэтому

$$w \leftarrow (2,1) + (-1,0) = (1,1)$$

и т.д. В итоге получим решение $w_1 = -1,\ w_2 = 3$.

Метод опорных векторов (SVM = Support Vector Machine)

Опишем самый популярный метод машинного обучения 1990х годов[1]. Для начала предполагаем, что в решаемой нами задаче бинарной классификации классы линейно разделимы. Рассмотрим различные линейные алгоритмы, см. рис. XB.3 – показан случай задачи с двумя признаками, здесь гиперплоскость является прямой.

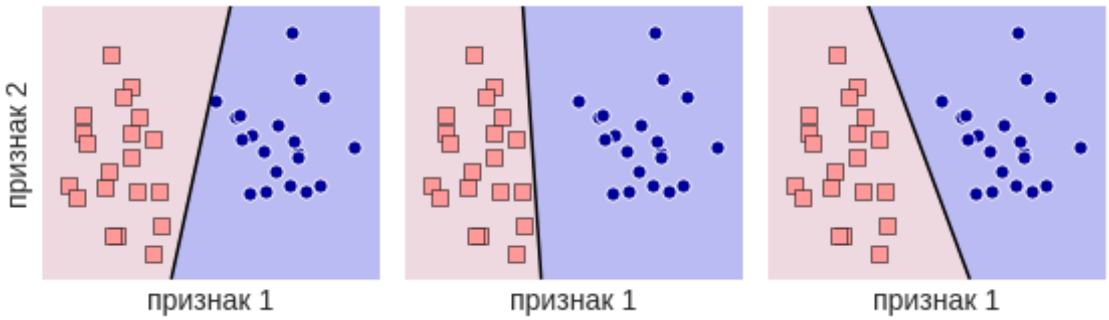


Рис. XB.3. Различные способы разделения классов гиперплоскостью.

До сих пор мы пытались просто разделить точки гиперплоскостью, но возникает естественный вопрос: «а какой линейный классификатор лучше?» Можно по построенным гиперплоскостям и конфигурациям точек посчитать расстояния от гиперплоскости до ближайших объектов двух классов. Чем больше сумма расстояний, тем лучше. Также расстояние до объекта класса +1 должно совпадать с расстоянием до объекта класса –1, чтобы гиперплоскость была как раз «посередине» между классами, см. рис. XB.4.

[1] Cortes C., Vapnik V. Support-vector networks //Machine learning. – 1995. – Т. 20. – С. 273-297.
Gunn S. R. et al. Support vector machines for classification and regression //ISIS technical report. – 1998. – Т. 14. – №. 1. – С. 5-16.
Burges C. J. C. A tutorial on support vector machines for pattern recognition //Data mining and knowledge discovery. – 1998. – Т. 2. – №. 2. – С. 121-167.

Указанная сумма расстояний визуально соответствует ширине разделяющей полосы между классами, которая получается параллельным переносом разделяющей гиперплоскости в пределах, пока гиперплоскость остаётся разделяющей. Аргументация в максимизации такой ширины и «срединного» расположения гиперплоскости простая: если мы немного ошибёмся с коэффициентами или сами объекты нам даны с небольшим ошибками (значения признаков зашумлены), то больше шансов, что построенная нами гиперплоскость всё-таки разделяет классы.

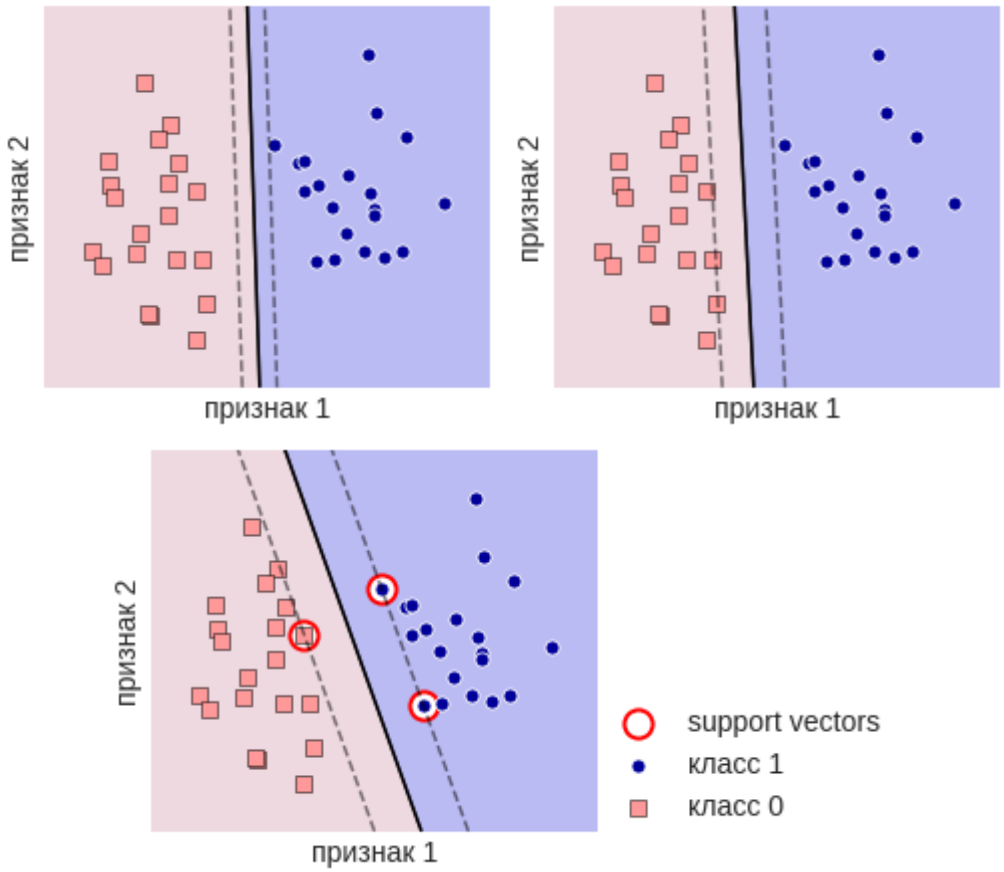


Рис. XB.4. Разделение классов полосами разной ширины.

Теперь формализуем описанное выше в виде постановки математической задачи, которая определяет метод SVM. Пусть дана обучающая выборка:

$$\{(x_i, y_i)\}_{i=1}^{m},$$

метки $y_i \in Y = \{\pm 1\}$, мы хотим разделить точки двух разных классов гиперплоскостью

$$a(x) = \text{sgn}(w^{\text{T}} x + b)$$

(здесь не вводим фиктивный константный признак). Таким образом, для объектов обучающей выборки должны выполняться неравенства:

$$\begin{cases} w^{\mathrm{T}} x_i + b > 0, & y_i = +1, \\ w^{\mathrm{T}} x_i + b < 0, & y_i = -1. \end{cases} \qquad \text{(ХВ.03)}$$

Заметим, что коэффициенты гиперплоскости нам нужны с точностью до множителя $\alpha > 0$, поскольку

$$\mathrm{sgn}(w^{\mathrm{T}} x + b) = \mathrm{sgn}(\alpha w^{\mathrm{T}} x + \alpha b),$$

поэтому можно нормировать коэффициенты. Можно также менять смещение b при условии сохранения неравенств. Если выполняются (ХВ.03), то

$$\begin{cases} w^{\mathrm{T}} x_i + b \geq +\varepsilon_{+1} > 0, & y_i = +1, \\ w^{\mathrm{T}} x_i + b \leq -\varepsilon_{-1} < 0, & y_i = -1. \end{cases}$$

Поделив неравенства на $\min(\varepsilon_{+1}, \varepsilon_{-1})$, приходим к выводу, что можно потребовать выполнение неравенств

$$w^{\mathrm{T}} x_i + b \geq +1 \text{ если } y_i = +1,$$
$$w^{\mathrm{T}} x_i + b \leq -1 \text{ если } y_i = -1$$

(это эквивалентно разделимости точек гиперплоскостью) и можно считать, что

$$\min_i | w^{\mathrm{T}} x_i + b | = 1. \qquad \text{(ХВ.04)}$$

Используем сокращённую форму записи получившейся системы:

$$y_i (w^{\mathrm{T}} x_i + b) \geq 1.$$

На рис. ХВ.5 показана гиперплоскость, которая удовлетворяет описанным требованиям. Обведены точки, для которых выполняется (ХВ.04) – в них происходит касание разделяющей полосы с выборкой.

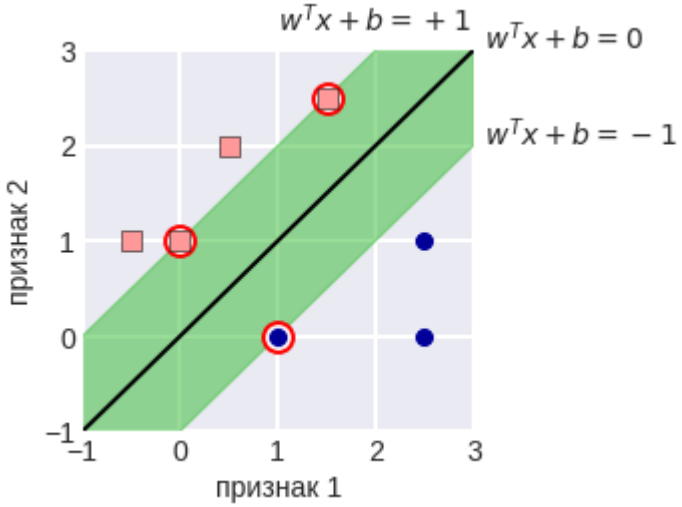


Рис. XB.5. Оптимальная разделяющая гиперплоскость.

Расстояние от точки x_i до гиперплоскости, определяемой уравнением $w^{\mathrm{T}}x+b=0$, как известно[1], равно

$$\rho(x_i, w^{\mathrm{T}}x+b)=\frac{|w^{\mathrm{T}}x_i+b|}{||w||}.$$

Мы хотим, чтобы удвоенный минимум из этих расстояний (это ширина полосы), который мы назовём **нормированным зазором[2] (margin)** был максимален, используя равенство (XB.04):

$$\min_i \frac{2|w^{\mathrm{T}}x_i+b|}{||w||}=\frac{2}{||w||}\to\max.$$

Немного переписав полученные выражения, возведя норму в квадрат для удобства оптимизации, получили **задачу квадратичного программирования (QP = Quadratic Program) с *m* ограничениями** (constraints):

$$\frac{||w||^2}{2}\to\min,$$

$$y_i(w^{\mathrm{T}}x_i+b)\geq 1,\ i\in\{1,2,\ldots,m\}.$$

[1] По идее, это доказывается в школьном курсе (для прямой и плоскости).

[2] До этого зазором мы называли выражение $z_t=y_t(w^{\mathrm{T}}x_t+b)$ (с учётом отсутствия фиктивного признака), сейчас мы рассматриваем выражение $|z_t|/||w||$ (модуль можно было бы и опустить из-за специфики решаемой задачи).

Линейный метод Hard Margin SVM – это линейный классификатор, параметры которого получаются в результате решения описанной задачи. Также, как при регуляризации линейной регрессии, здесь мы хотим квадрат нормы весов сделать меньше. Заметим, что решение полученной задачи существует и единственно, для задачи квадратичного программирования существует много солверов (программ и библиотек, позволяющих получать решения в конкретных задачах).

Мы получили геометрическое обоснование регрессии в линейном классификаторе: через максимизацию ширины разделяющей полосы.

Представим один из способов решения полученной задачи: **через решение двойственной задачи**. Для этого выписываем Лагранжиан (как принято в задачах с ограничениями):

$$\min_{w,b} \max_{\alpha \geq 0} L(w,b,\alpha),$$

где

$$L(w,b,\alpha) = \frac{w^{\mathrm{T}} w}{2} + \sum_{i=1}^{m} \alpha_i (1 - y_i (w^{\mathrm{T}} x_i + b)).$$

Возьмём производные (точнее по w это будет градиент), приравняем к нулю:

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{m} \alpha_i y_i x_i,$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^{m} \alpha_i y_i = 0.$$

Таким образом, оптимальный вектор весов – взвешенная сумма признаковых описаний объектов из обучения (если подумать, также происходит и при настройке персептрона, и в линейной и логистической регрессиях). Подставив полученные выражения в Лагранжиан, мы совершаем переход к двойственной задаче

$$\max_{\alpha \geq 0} \left[\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^{\mathrm{T}} x_j \right]$$

$$\text{при условиях } \sum_{i=1}^{m} \alpha_i y_i = 0.$$

Интересно, что в этой задаче информацию о признаковых описаниях объектов мы используем лишь в виде их попарных скалярных произведений! Когда решим двойственную задачу, то параметры линейного классификатора получаются по формулам:

$$w = \sum_{i=1}^{m} \alpha_i y_i x_i \qquad \text{(XB.05)}$$

(см. связь между решением прямой и обратной задачи выше),

$$b = -\frac{1}{2}(\min_{i:\, y_i=+1} w^{\mathrm{T}} x_i + \max_{i:\, y_i=-1} w^{\mathrm{T}} x_i)\,. \qquad \text{(XB.06)}$$

Здесь для нахождения b мы воспользовались геометрическими соображениями, что смещение соответствует гиперплоскости, которая находится в центре полосы, а края полосы как раз соответствуют усредняемым в (XB.06) значениям, см. рис. XB.5. Заметим, что большинство[1] коэффициентов α_i обратится в ноль из-за условий Кунна-Таккера, которые утверждают, что для решения справедливы равенства

$$\alpha_i(1 - y_i(w^{\mathrm{T}} x_i + b)) = 0\,,\ i \in \{1,2,\ldots,m\},$$

т.е. если $\alpha_i > 0$, то скобка обращается в ноль, тогда объект x_i называется **опорным (support vector)** и лежит на границе $y_i(w^{\mathrm{T}} x_i + b) = 1$ (эти объекты выделены на рис. XB.5). Заметим, что наше решение (XB.1) зависит только от опорных объектов (например, если остальные объекты удалить, то решение не изменится).

Оптимальная разделяющая гиперплоскость зависит только от объектов на границе разделения.

Полученную двойственную задачу можно переписать в матричном виде, поскольку

$$\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^{\mathrm{T}} x_j = \alpha^{\mathrm{T}} H \alpha$$

[1] Вообще говоря, нет гарантии, что большинство. Более того, можно привести примеры, когда таких коэффициентов не будет. Но в реальных задачах опорных объектов, как правило не так много. Далее будет рассмотрен метод Soft Margin SVM, в нём формально опорных объектов может быть существенно больше.

при $\alpha=(\alpha_1,\ldots,\alpha_m)^{\mathrm{T}}$ и $H=\| y_i y_j x_i^{\mathrm{T}} x_j \|_{m\times m}$, последняя матрица конгруэнтна[1] матрице Грама и является неотрицательно определённой. Решение двойственной задачи существует, но в отличие от прямой, вообще говоря, неединственное.

Итак, метод опорных векторов (SVM) с помощью решения двойственной задачи применяется следующим образом:

Запомним: в SVM нужны лишь попарные скалярные произведения признаковых описаний. В обучении – для пар объектов обучения, в тесте – для пар (объект теста, опорный объект).

1. Построить матрицу $H=\| y_i y_j x_i^{\mathrm{T}} x_j \|_{m\times m}$.

2. Решить задачу оптимизации

$$-\frac{1}{2}\alpha^{\mathrm{T}} H\alpha+\tilde{1}^{\mathrm{T}}\alpha\to\max$$

при условиях $\alpha\geq 0$, $y^{\mathrm{T}}\alpha=0$ (здесь везде векторная запись, $\tilde{1}=(1,\ldots,1)$).

3. Выразить параметры метода:

$$w=\sum_{i=1}^{m}\alpha_i y_i x_i\,,$$

$$S=\{i\in\{1,2,\ldots,m\}\,|\,\alpha_i>0\}\,,$$

$$b=\frac{1}{|S|}\sum_{i\in S}\left(y_i-w^{\mathrm{T}}x_i\right).$$

Здесь мы параллельно предложили альтернативный способ представления b. Заметим также, что

$$b=\frac{1}{|S|}\sum_{i\in S}\left(y_i-w^{\mathrm{T}}x_i\right)=\frac{1}{|S|}\sum_{i\in S}\left(y_i-\sum_{j\in S}\alpha_j y_j x_j^{\mathrm{T}}x_i\right),$$

а итоговый алгоритм имеет вид:

$$a(x)=\mathrm{sgn}(w^{\mathrm{T}}x+b)=\mathrm{sgn}(\sum_{i\in S}\alpha_i y_i x_i^{\mathrm{T}}x+b)\,.$$

Мы в формулах синим цветом выделили скалярные описания признаковых описаний: наше решение (коэффициенты гиперплоскости) и ответ алгоритма

[1] Представима в виде $H=P^{\mathrm{T}}GP$, где G – матрица Грама.

зависят только от скалярных произведений (а не от конкретных значений признаков).

Мы показали переход к двойственной задаче. Зачем его осуществлять, если можно решить исходную задача напрямую (ниже мы опишем даже оптимизационный алгоритм для решения более общей задачи)? Это может быть выгодно по следующим соображениям:

1. Из-за размерностей: если признаковое пространство большое, то удобнее решать двойственную задачу.

2. Двойственная задача записана в простой форме и также можно использовать готовые солверы.

3. В двойственной задаче вся информация об объектах передаётся в попарных скалярных произведениях их признаковых описаний. В дальнейшем это позволит обобщить метод (см. kernel tricks).

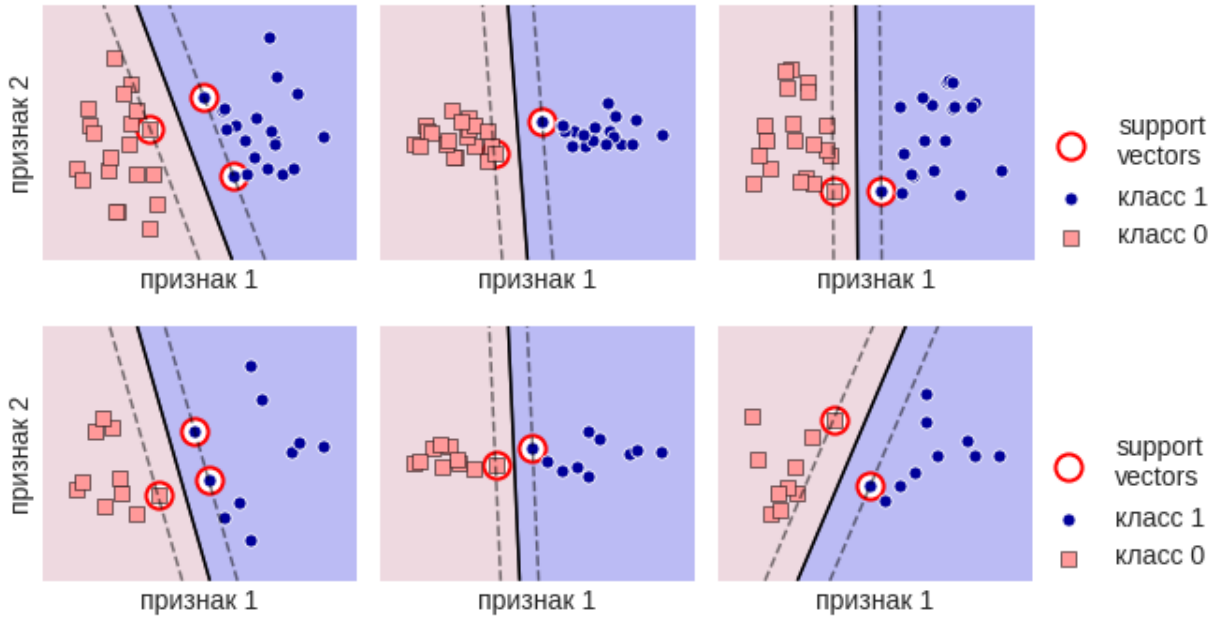


Рис. XB.6. Изменения оптимальной гиперплоскости при уменьшении масштаба одного из признаков (средний столбец) и добавлении случайного признака (правый столбец).

Отметим, что метод SVM мы вывели «из геометрических соображений» об оптимальной разделимости гиперплоскостью (оптимальность понимается как решение задачи максимизации ширины разделяющей полосы). Естественно, он довольно чувствителен к масштабированию признаков и наличию шумовых признаков. Ни рис. XB.6 в левом столбце показаны две модельные задачи, во втором столбце один из признаков сжимается в 10 раз (т.е. мы меняем масштаб)

– из-за чего меняется положение оптимальной разделяющей гиперплоскости, которую находит метод SVM, и опорные объекты, в третьем столбце один из признаков заменяется на случайный – также меняется гиперплоскость и опорные объекты.

Soft Margin SVM: разделение допуская ошибки

До настоящего момента мы предполагали линейную разделимость представителей разных классов в обучающей выборке задачи бинарной классификации. Теперь предположим, что её нет. В этом случае при использовании SVM можно применять два подхода, см. рис. XB.7:

1) разделять объекты гиперплоскостью, минимизируя число ошибок,

2) использовать нелинейные разделяющие поверхности.

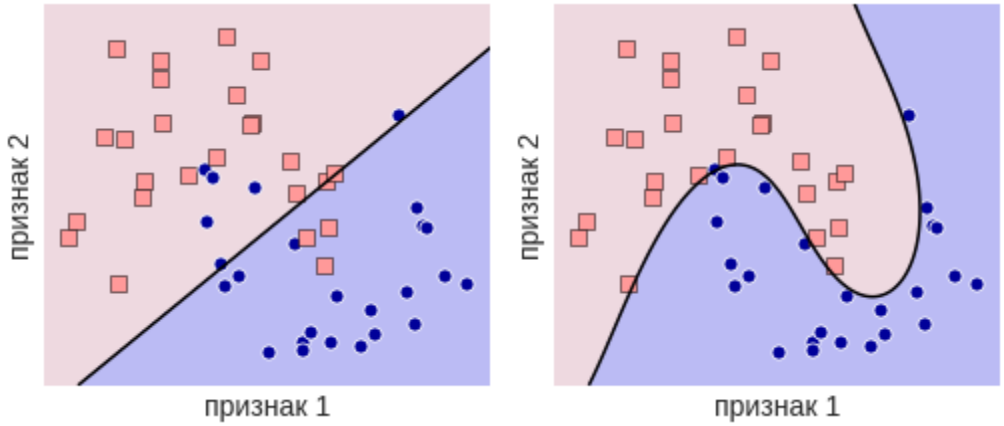


Рис. XB.7. Две стратегии решения нелинейных задач: решать оптимальным линейным методом (слева) и решать нелинейным методом (справа).

Второй переход потом подробно разберём позже, он соответствует переходу в другое признаковое пространство. Первый подход превращает SVM в **Soft Margin SVM**: в математической формулировке мы добавляем **штрафующие переменные (slack variables)**, которые «измеряют» величину заступа (насколько большая ошибка произошла на конкретном объекте – насколько соответствующая точка далеко от разделяющей гиперплоскости):

$$y_i(w^{\mathrm{T}}x_i + b) \geq 1 - \xi_i,\ \xi_i \geq 0,$$

см. также рис. XB.8: показана «разделяющая» гиперплоскость, но теперь она уже не разделяет объекты разных классов, соответствующая ей полоса,

несколько объектов, которые оказались «в чужом полупространстве». Для этих объектов и будут положительные заступы $\xi_i > 0$.

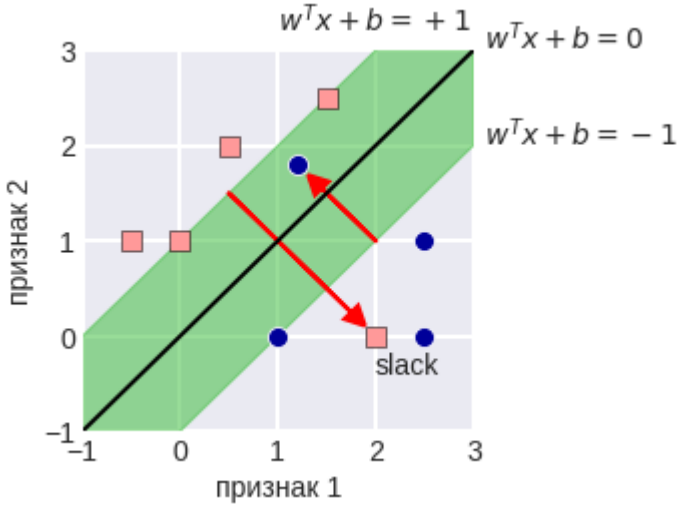


Рис. XB.8. Разделение с ошибками.

Мы позволяем объектам «залезать» в полупростанство другого класса, но не хотим, чтобы было много больших «залезаний», поэтому прямая задача записывается в виде

$$\frac{\|w\|^2}{2} + C\sum_{i=1}^{m}\xi_i \to \min$$

$$y_i(w^{\mathrm{T}}x_i + b) \geq 1 - \xi_i,\ \xi_i \geq 0,\ i \in \{1,2,\ldots,m\}.$$

Можно и по-другому формализовать нежелание много и грубо ошибаться, но при рассматриваемой формализации не сильно усложняется задача QP.

Это тоже задача квадратичного программирования (QP), но в ней два раза больше ограничений, по сравнению с обычным SVM. Коэффициент C нужен для регулирования баланса между оптимизацией зазора и ошибки на обучении. Если $C = 0$, то получим нулевое решение $w = (0,\ldots,0)^{\mathrm{T}}$, если $C \to +\infty$, то получим решение как в обычном SVM, который теперь логично назвать **«hard margin objective SVM»**.

Двойственная задача (вывод оставляем читателю, он аналогичен проведённому ранее) записывается так:

$$\sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^{\mathrm{T}}x_j \to \max_{0\leq\alpha\leq C}$$

$$\sum_{i=1}^{m}\alpha_i y_i = 0.$$

Таким образом, появляется лишь новое ограничение $\alpha \le C$. В остальном, двойственная задача совпадает с двойственной задачей для SVM.

Итак, метод опорных векторов Soft-Margin SVM в линейно неразделимом случае является линейным классификатором, параметры которого можно получить следующим алгоритмом.

1. Построить матрицу $H = \| y_i y_j x_i^{\mathrm{T}} x_j \|_{m \times m}$.

2. Решить задачу оптимизации

$$-\frac{1}{2}\alpha^{\mathrm{T}} H \alpha + \tilde{1}^{\mathrm{T}} \alpha \to \max$$

при условиях

$$0 \le \alpha \le C,\ y^{\mathrm{T}} \alpha = 0$$

(здесь везде векторная запись).

3. Получить решение

$$w = \sum_{i=1}^{m} \alpha_i y_i x_i\,,$$

$$S = \{i \in \{1,2,\ldots,m\} \mid 0 < \alpha_i \le C\}\,,$$

$$b = \frac{1}{|S|} \sum_{i \in S} \left(y_i - w^{\mathrm{T}} x_i \right).$$

Здесь красным цветом в формулах выделено отличие от Hard Margin SVM. Итоговый алгоритм, как и раньше, имеет вид:

$$a(x) = \mathrm{sgn}(w^{\mathrm{T}} x + b) = \mathrm{sgn}\left(\sum_{i \in S} \alpha_i y_i x_i^{\mathrm{T}} x + b\right).$$

Теперь предложим **численное решение Soft-Margin SVM**. Преобразуем ограничения:

$$\begin{cases} \xi_i \ge 1 - y_i (w^{\mathrm{T}} x_i + b) \\ \xi_i \ge 0 \end{cases} \quad \Leftrightarrow \quad \xi_i \ge \max[1 - y_i (w^{\mathrm{T}} x_i + b), 0],$$

т.к. в функционал для минимизации

$$\frac{\|w\|^2}{2} + C\sum_{i=1}^{m}\xi_i \to \min$$

входит сумма ξ_i, то берём минимально возможные слагаемые:

$$\xi_i = \max[1 - y_i(w^{\mathrm{T}}x_i + b), 0].$$

В результате получили следующую задачу оптимизации:

$$\frac{1}{2C}\|w\|^2 + \sum_{i=1}^{m}\max[1 - y_i(w^{\mathrm{T}}x_i + b), 0] \to \min, \qquad \text{(XB.07)}$$

в которой оптимизируемый функционал логически разбивается на L2-регуляризационное слагаемое (выделено красным) и на одну из оценок 0-1-loss, которую назовём **Hinge Loss** (выделено синим), см. также рис. XB.9.

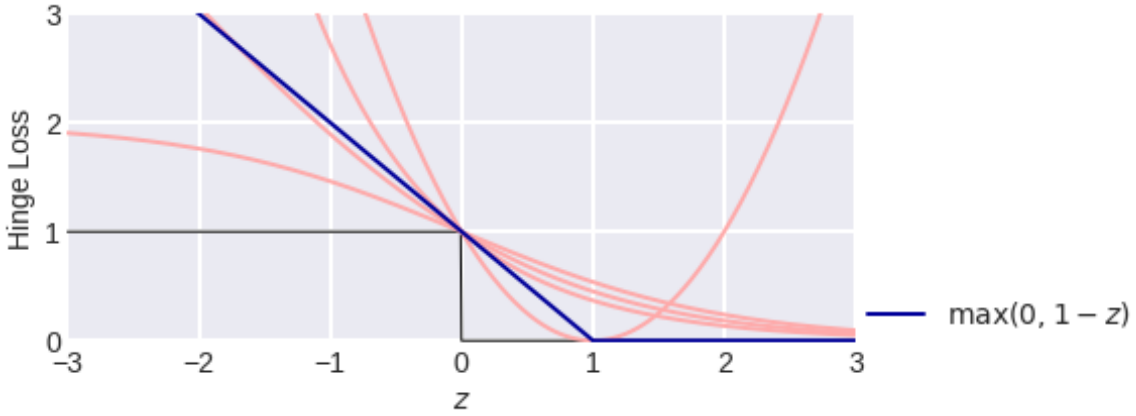
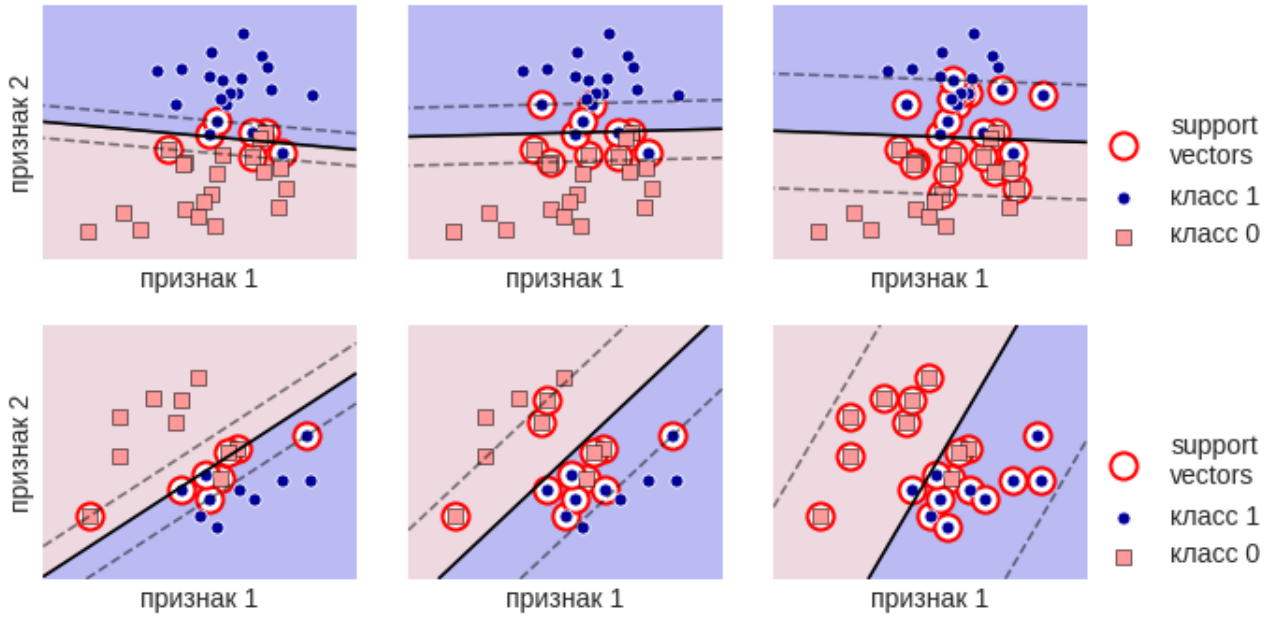


Рис. XB.9. Hinge loss.

Этот вывод объясняет, почему во многих реализациях SVM (и многих других линейных методов классификации) в качестве гиперпараметра используется не коэффициент регуляризации (при квадрате нормы), а обратный к нему (т.е. C): этот параметр получился в результате формализации задачи построения оптимальной разделяющей гиперплоскости и вывода её решения, имеет смысл «баланса» между задачами увеличения ширины полосы и уменьшения величин заступов, и имеет «историческое» обозначение.

Из формулы (XB.07) видно, что если у объекта большой заступ (зазор на объекте $y_i(w^{\mathrm{T}}x_i + b)$ больше 1), то он не влияет на решение (соответствующее ему слагаемое нулевое). На рис. XB.10 отмечены опорные объекты (это те, которые метод SVM в библиотеке sklearn выдаёт как опорные) – это все объекты с зазором не превосходящим 1. Заметим, что здесь уже теряется интерпретация опорных объектов, как объектов, лежащих на границе полосы.



$C = 1.0$ $C = 0.1$ $C = 0.01$

Рис. XB.10. Soft-Margin SVM при разной регуляризации.

Метод опорных векторов для регрессии[1] (SVM Regression)

Существует также регрессионный вариант SVM, в нём формализуется желание как можно лучше решить задачу регрессии линейным методом с ε-точностью:

$$\frac{\|w\|^2}{2} \to \min \text{ (регуляризация)},$$

$$|w^{\mathrm{T}} x_i + b - y_i| \le \varepsilon \text{ (небольшие ошибки)},$$

$i \in \{1,2,\ldots,m\}$. Геометрически выполнение описанного выше неравенства для точки означает попадание её в полосу (ширины 2ε, с центром на гиперплоскости, если ширину измерять по оси y), см. рис. XB.11. Поскольку нет надежды, что удастся найти решение, при котором все объекты обучающей выборки попадают в 2ε-полосу, мы не будем требовать строгого выполнения описанных неравенств, а будем штрафовать за их невыполнение.

[1] В главе про линейную классификацию мы возвращаемся к задаче регрессии, поскольку больше не будет повода рассказать об этом методе.

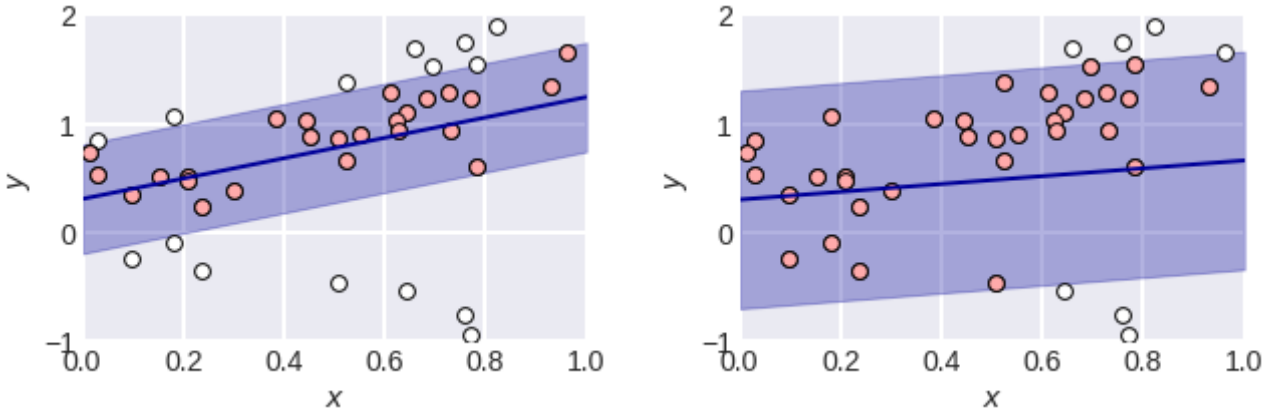


Рис. XB.11. Регрессионное решение и полоса ε-точности: $\varepsilon = 0.5$ – слева, $\varepsilon = 1$ – справа.

Запишем такую функцию ошибки:

$$\frac{\|w\|^2}{2} + C\sum_{i=1}^{m} \max[|w^{\mathrm{T}} x_i + b - y_i| - \varepsilon, 0] \to \min,$$

которую можно проинтерпретировать как сумму регуляризатора и функции ошибки $\max[|z| - \varepsilon, 0]$. Это т.н. **$\varepsilon$-чувствительная ошибка**, график которой показан на рис. XB.12: если мы ошибаемся на каком-то объекте меньше, чем на ε, тогда штраф за ошибку нулевой, иначе он линейно зависит от $|w^{\mathrm{T}} x_i + b - y_i|$.

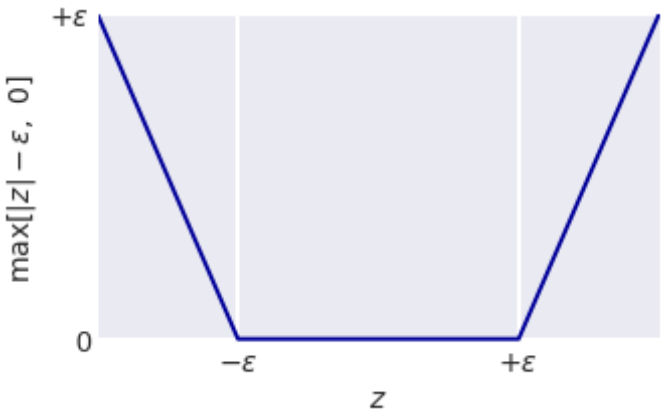


Рис. XB.12. График ε-чувствительной ошибки.

После замены переменных задача сводится к задаче квадратичного программирования (QP). Продемонстрируем переход к двойственной задаче. Для неравенства

$$|w^{\mathrm{T}} x_i + b - y_i| \le \varepsilon$$

введём штрафующие слагаемые $\xi_i^+ \ge 0, \xi_i^- \ge 0, i \in \{1, 2, \ldots, m\}$:

$$w^{\mathrm{T}} x_i + b - y_i \le +\varepsilon + \xi_i^+,$$

$$w^{\mathrm{T}} x_i + b - y_i \ge -\varepsilon - \xi_i^-,$$

тогда наш функционал для оптимизации перепишется как

$$\frac{\| w \|^2}{2} + C\sum_{i=1}^{m}(\xi_i^+ + \xi_i^-) \to \min.$$

Теперь понятно, почему мы назвали слагаемые штрафующими: сумма $\xi_i^+ + \xi_i^-$ в точности равна ε-чувствительной ошибке на i-м объекте, при этом одно слагаемое штрафует за залезание в одну сторону от 2ε-полосы, другое – за залезание в другую сторону. С их помощью мы избавились от модуля, но увеличили число переменных и ограничений. Дальше переходим к Лагранжиану (не будем это подробно описывать, мы делали это для Hard Margin SMV Classification). После перехода к двойственной задаче получаем

$$\sum_{i=1}^{m}(\alpha_i^+ - \alpha_i^-)(w^{\mathrm{T}} x_i + b - y_i) - \varepsilon\sum_{i=1}^{m}(\alpha_i^+ - \alpha_i^-) - \frac{1}{2}\sum_{i,\,j}(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)x_i^{\mathrm{T}} x_j \to \max$$

$$0 < \alpha_i^+ \le C, 0 < \alpha_j^+ \le C, \sum_{i=1}^{m}(\alpha_i^+ - \alpha_i^-) = 0$$

После решения этой задачи квадратичного программирования искомые параметры определяются по формулам:

$$w = \sum_{i=1}^{m}(\alpha_i^+ - \alpha_i^-)x_i,$$

$$S = \{i | 0 < \alpha_i^+ \le C, 0 < \alpha_i^- \le C, (\xi_i^+ = 0) \vee (\xi_i^- = 0)\},$$

$$b = \frac{1}{|S|}\sum_{i \in S}\left(y_i - w^{\mathrm{T}} x_i - \varepsilon\right).$$

Вместо решения двойственной задачи можно и напрямую минимизировать сумму ε-чувствительной ошибки и регуляризатора:

$$\sum_{i=1}^{m}\max[|\, w^{\mathrm{T}} x_i + b - y_i \,| - \varepsilon, 0] + \frac{1}{2C}\| w \|^2 \to \min.$$

Линейный SVM-классификатор: итоги

1. Используется естественное определение «оптимальной разделяющей гиперплоскости» (здесь ясная геометрическая интерпретация и есть некоторая защита от проклятия размерности), *как мы увидим дальше, в нелинейном SVM такая интерпретация уже немного теряется.*

2. Есть теоретическое обоснование, что большой зазор соответствует меньшему переобучению (мы, правда, не рассказывали о соответствующих теоремах), но это видно по обоснованной связи с L2-регуляризацией,

3. При оптимизации нет проблем локальных минимумов (минимум единственный в прямой задаче),

4. Решение определяется «опорными объектами» (их сложнее всего классифицировать, т.к. они лежат на границе класса – в Hard Margin SVM или у них «не очень большой зазор» – в Soft Margin SVM), только их можно оставить в выборке, т.к. от остальных объектов решение не зависит. *Но за это и критиковали SVM: странно в задаче диагностики болезни строить решение, анализируя самых здоровых больных и самых больных здоровых, а не «типичных здоровых» и «типичных больных».*

5. Есть нелинейные модификации «kernel tricks» для метода SVM, которые мы изучим дальше (они превращают SVM в нелинейный метод).

6. На практике при применении SVM (как и других линейных методов) должно быть хорошее пространство (однородные признаки в одной шкале), тогда успешно работают линейные SVM (*нелинейные – с ядрами – успешно заменяются другими алгоритмами*).

7. Метод SVM не подходит для больших данных, особенно нелинейный SVM (будет дальше), т.к. приходится решать двойственную задачу с $m \times m$-матрицей H. Также плохо решаются задачи с зашумлёнными данными.

Логистическая регрессия

Пусть дана задача бинарной классификации: $Y = \{0,1\}$ с вещественными признаками: $X = \mathbb{R}^n$. Мы хотим получать оценки принадлежности к классу 1:

$$a(x) \in [0,1],$$

их можно интерпретировать как вероятности принадлежности к классу 1, но корректность такой интерпретации обсудим позже. Допустим, мы хотим использовать линейную модель. Любая линейная функция на $\mathbb{R}^n$ будет получать значения в $\mathbb{R}$, поэтому нужна **деформация (transfer function)**:

$$\sigma : \mathbb{R} \to [0,1].$$

Оценки принадлежности к классу 1 будет получаться по формуле $\sigma(w^{\mathrm{T}}x)$, т.е. с помощью проекции в одномерное пространство и последующей деформации (если пространство не пополнено фиктивным константным признаком, то проекцию следует записать как $w^{\mathrm{T}}x + w_0$).

Обычно используются два вида деформации, в логистической регрессии – **логистическая функция (сигмоида)**:

$$\sigma(z) = \frac{1}{1+e^{-z}},$$

в Probit-регрессии – **интегральная функция нормального стандартного распределения (Normal Cumulative Distribution Function)**:

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} \exp(-t^2 / 2)\partial t,$$

графики этих функций показаны на рис. XB.13.

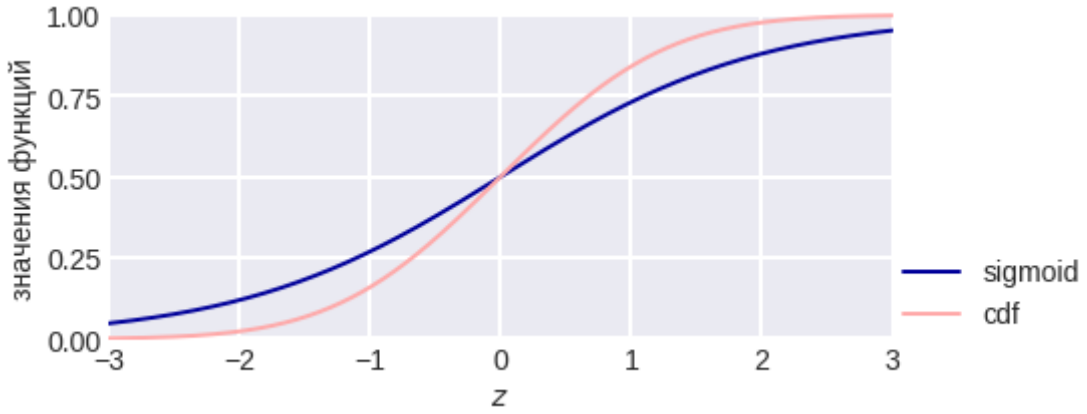


Рис. XB.13. Функции деформации.

В алгоритме **логистическая регрессия**[1] оценку вероятности принадлежности к классу 1 объекта $x=(X_1,\ldots,X_n)$ получают с помощью логистической функции деформации:

$$P(y=1|x)=\sigma(z)=\frac{1}{1+e^{-z}}\in(0,1),$$

значение

$$z=z(x)=w_0+w_1X_1+\ldots+w_nX_n.$$

называют **логитом**, а обратную функцию к сигмоиде (это монотонное преобразование) называют **логит-преобразованием (logit-transformation)**:

$$\log\left(\frac{\sigma(z)}{1-\sigma(z)}\right)=z.$$

Отметим несколько интересных свойств логистической функции. Во-первых,

$$\sigma(z)+\sigma(-z)=\frac{1}{1+e^{-z}}+\frac{1}{1+e^{+z}}=\frac{e^{+z}}{1+e^{+z}}+\frac{1}{1+e^{+z}}=1$$

Поскольку вероятность принадлежности классу 1 $P(y=1|x)$, оценивают с помощью $\sigma(z)$, вероятность принадлежности классу 0 $P(y=0|x)$ оценивают с помощью

$$P(Y=0|x)=1-\sigma(z)=\sigma(-z),$$

т.е. «сменой знака у логита».

Во-вторых, производная сигмоиды выражается через саму функцию

$$\frac{\partial\sigma(z)}{\partial z}=\left(\frac{1}{1+e^{-z}}\right)'=\frac{-(-e^{-z})}{(1+e^{-z})^2}=\frac{1+e^{-z}-1}{(1+e^{-z})^2}=\frac{1}{1+e^{-z}}\left(1-\frac{1}{1+e^{-z}}\right)=\sigma(z)(1-\sigma(z))$$

(нам пока это свойство не понадобится).

На рис. XB.14 показан геометрический смысл логистической регрессии. Уравнение $\sigma(x)=0.5$ (или $z(x)=0$) задаёт разделяющую гиперплоскость. Здесь мы задали порог 0.5 для бинаризации вероятности, но это отдельный вопрос: как выбирать порог бинаризации? На прямой, ортогональной к этой

[1] Не смотря на название, логистическая регрессия используется для решения задачи классификации.

гиперплоскости, оценка меняется как раз по сигмоиде, см. рис. ХВ.15. Если спроецировать точки выборки на прямую, то при обучении метод старается получить большие значения $\sigma(x)$ для объектов класса 1 и маленькие для объектов класса 0. На рис. ХВ.15 сигмоида «прижимается» к этим объектам.

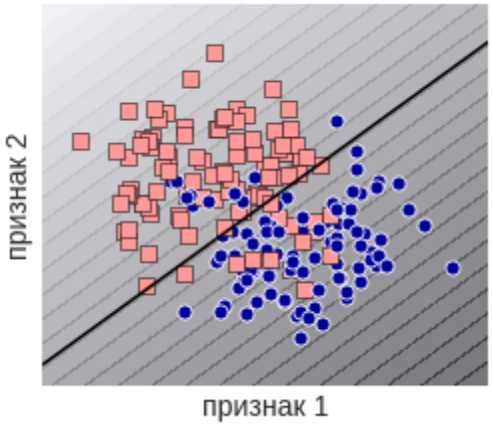


Рис. ХВ.14. Линии уровня логистической регрессии.

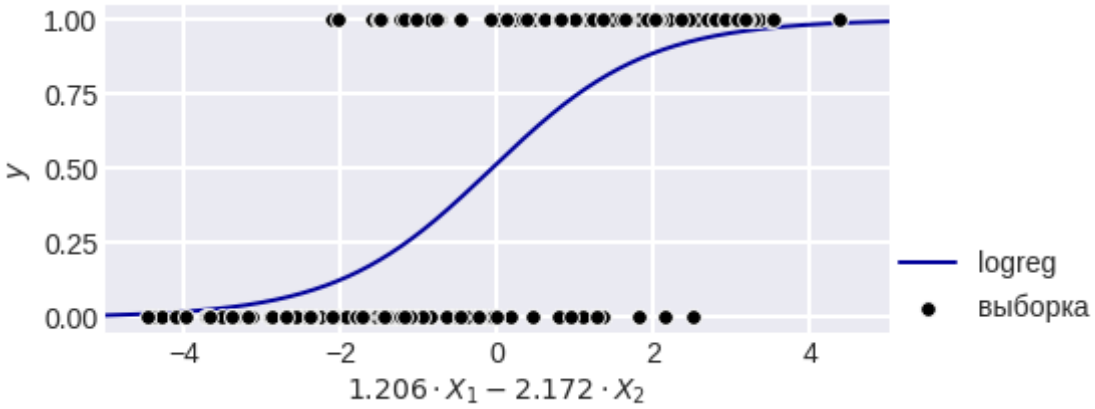


Рис. ХВ.15. Линии уровня логистической регрессии на прямой ортогональной к разделяющей поверхности.

На рис. ХВ.16 показаны значения сигмоиды от различных линейных функций, в выражении $\sigma(\alpha x + \beta)$ коэффициент α отвечает за крутизну, а β – за горизонтальное смещение.

Если не деформировать линейную регрессию сигмоидой, а использовать «как есть», то ответы алгоритма нельзя будет интерпретировать как вероятности, см. рис. ХВ.17. Напрашивается привести её ответы на отрезок [0, 1], но, как мы покажем дальше, деформация сигмоидой обосновывается при некоторых предположениях.

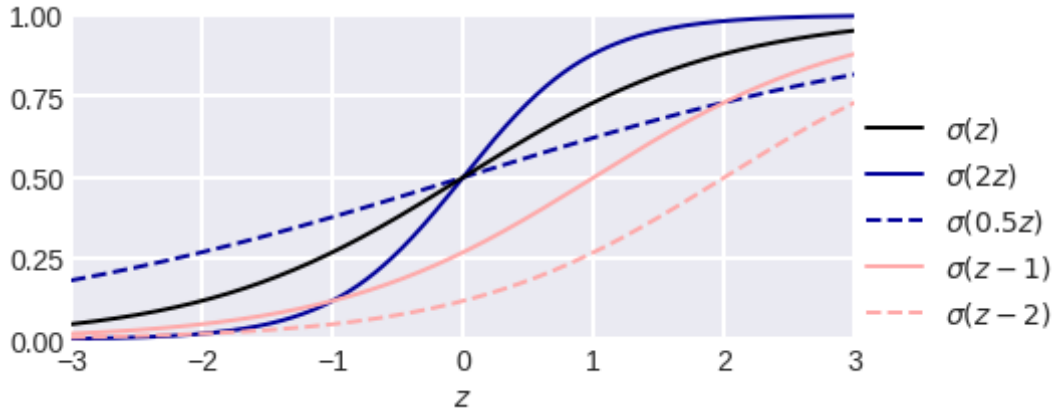


Рис. XB.16. Сигмоиды от линейных функций.

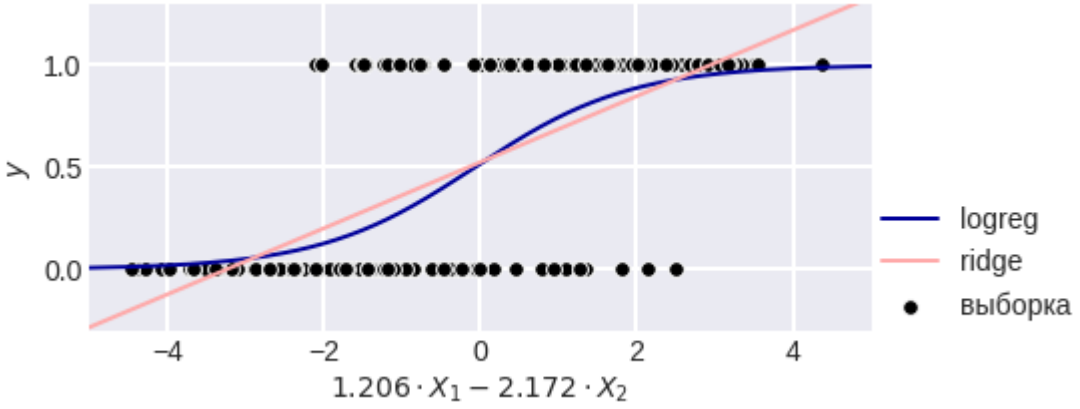


Рис. XB.17. Линии уровня логистической регрессии и линейной регресси на прямой ортогональной к разделяющей поверхности.

Действительно, пусть объекты в классах 0 и 1 распределены нормально с одинаковыми матрицами ковариации:

$$p(x \mid y=t) = \frac{1}{(2\pi)^{n/2} \mid \Sigma \mid^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_t)^{\mathrm{T}} \Sigma^{-1} (x-\mu_t) \right).$$

На рис. XB.18 изображена одномерная модельная задача с нормально распределёнными классами (плотности похожи на «холмики одинаковой ширины»).

Воспользуемся формулой Байеса для оценки вероятности принадлежности объекта к классу

$$p(y=t \mid x) = \frac{p(x \mid y=t)\, p(y=t)}{p(x \mid y=0)\, p(y=0) + p(x \mid y=1)\, p(y=1)}$$

(подробнее будет в главе «Байесовский подход»). При условии равновероятных классов, получаем

$$p(y=t\,|\,x)=\frac{\frac{1}{(2\pi)^{n/2}\,|\,\Sigma\,|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu_t)^{\mathrm{T}}\Sigma^{-1}(x-\mu_t)\right)}{\sum_t\frac{1}{(2\pi)^{n/2}\,|\,\Sigma\,|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu_t)^{\mathrm{T}}\Sigma^{-1}(x-\mu_t)\right)}=$$

$$=\frac{1}{1+\exp\left(+\frac{1}{2}(x-\mu_t)^{\mathrm{T}}\Sigma^{-1}(x-\mu_t)-\frac{1}{2}(x-\mu_{1-t})^{\mathrm{T}}\Sigma^{-1}(x-\mu_{1-t})\right)}=$$

$$=\sigma\left(-\frac{1}{2}\mu_t^{\mathrm{T}}\Sigma^{-1}x-\frac{1}{2}x^{\mathrm{T}}\Sigma^{-1}\mu_t+\frac{1}{2}\mu_t^{\mathrm{T}}\Sigma^{-1}\mu_t+\frac{1}{2}\mu_{1-t}{}^{\mathrm{T}}\Sigma^{-1}x+\frac{1}{2}x^{\mathrm{T}}\Sigma^{-1}\mu_{1-t}-\frac{1}{2}\mu_{1-t}{}^{\mathrm{T}}\Sigma^{-1}\mu_{1-t}\right)$$

Если ввести подходящие обозначения, то последнее выражение переписывается в виде $\sigma(w^{\mathrm{T}}x+w_0)$, т.е. вероятность выражается так, как оценка в логистической регрессии. Итак, искать решение в виде логистической регрессии разумно при

1) нормально распределённых классах (от этого допущения, к сожалению, нельзя избавиться, хотя при некоторых других распределениях также можно прийти к деформации-сигмоиде),

2) равенстве матриц ковариаций (если матрицы будут не равны, то сигмоида будет не от линейной функции, но используя преобразование признакового пространства – подробнее в главе про нелинейные методы – опять приходим к логистической регрессии),

3) равенстве априорных вероятностей классов $p(y=t)$ (можно ли избавиться от этого предположения – вынесено в задачу).

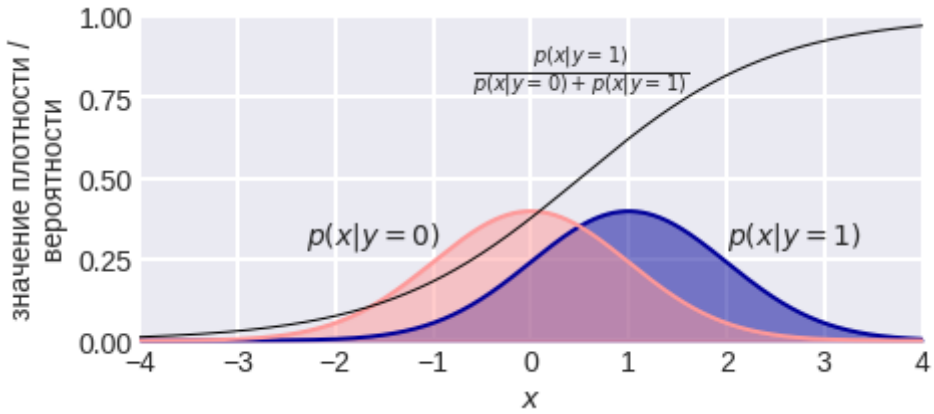


Рис. XB.18. Распределения объектов из двух классов в модельной задаче.

Попробуем обучить логистическую регрессию, воспользовавшись методом максимального правдоподобия:

$$L(w_0,\ldots,w_n)=\prod_{i:y_i=1}\sigma(w^{\mathrm{T}}x_i)\prod_{i:y_i=0}(1-\sigma(w^{\mathrm{T}}x_i))\to\max,$$

после логарифмирования получаем

$$\log L=\sum_{i:y_i=1}\log(\sigma(w^{\mathrm{T}}x_i))+\sum_{i:y_i=0}\log(\sigma(-w^{\mathrm{T}}x_i))\to\max.$$

Для удобства записи перейдём к меткам классов ±1: $y_i'=2y_i-1$, тогда последнее выражение запишется как

$$\log L=\sum_i\log(\sigma(y_i'w^{\mathrm{T}}x_i))=-\sum_i\log(1+\exp(-y_i'w^{\mathrm{T}}x_i))$$

Вычислим градиент:

$$\nabla_w\log L=-\sum_i\nabla_w\log(1+\exp(-y_i'w^{\mathrm{T}}x_i))=-\sum_i\frac{\exp(-y_i'w^{\mathrm{T}}x_i)}{1+\exp(-y_i'w^{\mathrm{T}}x_i)}(-y_i'x_i)=$$

$$=\sum_i\frac{y_i'x_i}{1+\exp(+y_i'w^{\mathrm{T}}x_i)}=\sum_i\sigma(-y_i'w^{\mathrm{T}}x_i)y_i'x_i$$

Линии уровней функции правдоподобия (точнее логарифма от этой функции) показаны на рис. ХВ.19. Здесь получается выпуклая задача оптимизации, как и в линейной регрессии.

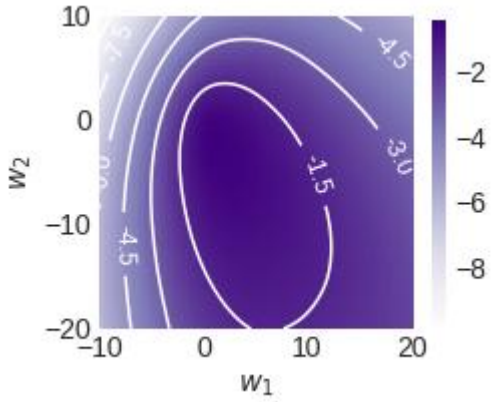


Рис. ХВ.19. Линии уровня логарифма правдоподобия.

В главе «Метрики качества» мы обратим внимание на то, что логарифм правдоподобия здесь соответствует специальной функции ошибки, которая

называется **логистической (LogLoss)**. Метод стохастического градиента (SGD) итерационно меняет веса по формуле

$$w \leftarrow w + \eta\sigma(-y_i' w^{\mathrm{T}} x_i) y_i' x_i$$

(здесь градиент прибавляется, а не вычитается, так как мы максимизируем, а не минимизируем правдоподобие). Вспомним формулу изменения весов в персептронном алгоритме, она может быть записана в виде

$$w \leftarrow w + \eta I[-y_i w^{\mathrm{T}} x_i \geq 0] y_i' x_i .$$

Эти две формулы очень похожи[1]: в первой сигмоида является сглаженным аналогом пороговой функции из второй формулы. Если при обучении персептрона пороговая функция активирует изменение при наличии ошибки (когда зазор $y_i w^{\mathrm{T}} x_i$ отрицательный), то при обучении логистической регрессии сигмоида оценивает величину ошибки / верность ответа: при маленьком зазоре значение сигмоиды будет близко к 1, при большом – к 0 и производится маленькая коррекция параметров (но параметры корректируются в любом случае). Также отметим, что персептронный алгоритм может получить любую разделяющую гиперплоскость, а в логистической регрессии ответ однозначно определён.

Рассмотрим обобщение логистической регрессии на случай **нескольких непересекающихся классов: многоклассовую логистическую регрессию (Multiclass Logistic Regression / Multinomial Regression)**. Пусть для j-го класса линейно получается оценка принадлежности к нему: $w_j^{\mathrm{T}} x$, вектор оценок надо перевести в вектор, который можно интерпретировать как распределение по классам:

$$(w_1^{\mathrm{T}} x, w_2^{\mathrm{T}} x, \ldots, w_l^{\mathrm{T}} x) \in \mathbb{R}^l \rightarrow (p_1(x), p_2(x), \ldots, p_l(x)) \in [0,1]^l_{\Sigma=1} .$$

Для этого подходит функция функция softmax, которая как раз любой вещественный вектор $(a_1, \ldots, a_l)$ «превращает» в распределение:

$$\mathrm{softmax}(a_1, \ldots, a_l) = \frac{1}{\exp(a_1) + \ldots + \exp(a_l)} [\exp(a_1), \ldots, \exp(a_l)],$$

получается вещественный вектор с положительными значениями сумма которых равна 1. Эту функцию также можно вывести из формулы Байеса, как

[1] Смотри также формулу для коррекции весов при обучении линейной регрессии.

мы это сделали с сигмоидой. Таким образом, в многоклассовой логистической регрессии при классификации объекта $x=(X_1,\ldots,X_n)$ вероятности оцениваются по формуле

$$P(y=k\,|\,x)\sim\frac{\exp(w_{0k}+w_{1k}X_1+\ldots+w_{nk}X_n)}{\sum_{j=1}^{l}\exp(w_{0j}+w_{1j}X_1+\ldots+w_{nj}X_n)}.$$

В задаче с пересекающимися классами логично использовать логистическую регрессию для каждого класса, определяя, принадлежит объект классу или нет:

$$(w_1^{\mathrm{T}}x,w_2^{\mathrm{T}}x,\ldots,w_l^{\mathrm{T}}x)\in\mathbb{R}^l\to(\sigma(w_1^{\mathrm{T}}x),\sigma(w_2^{\mathrm{T}}x),\ldots,\sigma(w_l^{\mathrm{T}}x))\in[0,1]^l.$$

Линейный дискриминант Фишера

Рассмотрим ещё один линейный метод, который предлагает немного по-другому взглянуть на линейное решение. Рассмотрим случай двух классов:

$$X=\mathbb{R}^n,\ Y=\{\pm1\},$$

$$X_\alpha=\{x_i\,|\,y_i=\alpha\},$$

$$m_\alpha=|\,X_\alpha\,|,$$

$$m=m_{+1}+m_{-1}.$$

Мы хотим все точки проецировать на прямую: $x\to w^{\mathrm{T}}x$, тогда проекция центра класса переходит в центр проекций объектов класса:

$$\mu_\alpha=\frac{1}{m_\alpha}\sum_{x_i\in X_\alpha}x_i\to w^{\mathrm{T}}\mu_\alpha=\frac{1}{m_\alpha}\sum_{x_i\in X_\alpha}w^{\mathrm{T}}x_i$$

«Дисперсии» (в ненормируемом виде) проекций запишутся как

$$\sigma_\alpha^2=\sum_{x_i\in X_\alpha}(w^{\mathrm{T}}x_i-w^{\mathrm{T}}m_\alpha)^2.$$

Какую прямую для проекции выбрать? Давайте выберем ту, на которой центры классов максимально удалены друг от друга, правда, при этом лучше удалённость оценивать не абсолютным расстоянием, а относительным

Надо учитывать не абсолютную разнесённость, а относительнyю.

(учитывая разброс внутри классов):

$$\frac{(w^{\mathrm{T}}\mu_{+1} - w^{\mathrm{T}}\mu_{-1})^2}{\sigma_{+1}^2 + \sigma_{-1}^2} \to \max .$$

Числитель оптимизируемого выражения можно переписать как

$$(w^{\mathrm{T}}\mu_{+1} - w^{\mathrm{T}}\mu_{-1})^2 = (w^{\mathrm{T}}(\mu_{+1} - \mu_{-1}))^2 =$$

$$= w^{\mathrm{T}}(\mu_{+1} - \mu_{-1})(\mu_{+1} - \mu_{-1})^{\mathrm{T}} w = w^{\mathrm{T}} S w,$$

здесь мы ввели матрицу $S = (\mu_{+1} - \mu_{-1})(\mu_{+1} - \mu_{-1})^{\mathrm{T}}$, которую называют **матрицей междуклассового разброса (between-class scatter)**, а слагаемые в знаменателе в виде

$$\sigma_\alpha^2 = \sum_{x_i \in X_\alpha} (w^{\mathrm{T}} x_i - w^{\mathrm{T}}\mu_\alpha)^2 = w^{\mathrm{T}} \sum_{x_i \in X_\alpha} (x_i - \mu_\alpha)(x_i - \mu_\alpha)^{\mathrm{T}} w =$$

$$= w^{\mathrm{T}} S_\alpha w,$$

мы также ввели матрицу **внутриклассового разброса (within-class scatter matrices)**

$$S_\alpha = \sum_{x_i \in X_\alpha} (x_i - \mu_\alpha)(x_i - \mu_\alpha)^{\mathrm{T}} .$$

Теперь наша оптимизационная задача переписалась в виде

$$\frac{w^{\mathrm{T}} S w}{w^{\mathrm{T}}(S_{+1} + S_{-1})w} \to \max .$$

Если вычислить градиент и приравнять к нулю, то получим

$$w^{\mathrm{T}} S w \cdot (S_{+1} + S_{-1})w = w^{\mathrm{T}}(S_{+1} + S_{-1})w \cdot S w,$$

здесь синим в формулах выделены выражения $w^{\mathrm{T}} S w$, $w^{\mathrm{T}}(S_{+1} + S_{-1})w$, которые являются скалярами, а красным – $S w$ – выражение, которое равно вектору $(\mu_{+1} - \mu_{-1})$, умноженному на скаляр. Поэтому наше выражение упрощается как

$$\mathrm{const} \cdot (S_{+1} + S_{-1})w = \mathrm{const} \cdot (\mu_{+1} - \mu_{-1}) .$$

Нас интересует только направление вектора w, а не его величина, поэтому можно считать решением

Полезный приём, когда вектор нужен с точностью до направления – «заединичить» константы.

$$w \propto (S_{+1} + S_{-1})^{-1}(\mu_2 - \mu_1).$$

Можно было провести оптимизацию и по-другому. Если вектор w умножить на константу, то числитель и знаменатель сокращаются, поэтому можно изначально задаться условием $w^{\mathrm{T}}(S_{+1} + S_{-1})w = 1$.

На рис. XB.20-21 показаны выборки, отмечены центры классов, чёрным цветом – построенные прямые (для проекции), концентрические эллипсы соответствуют матрицам, точнее являются линиями уровней нормальных распределений с такими матрицами ковариации $S_{+1}, S_{-1}, S_{+1} + S_{-1}$, матрицы $S_{+1} + S_{-1}$ показаны зелёным цветом.

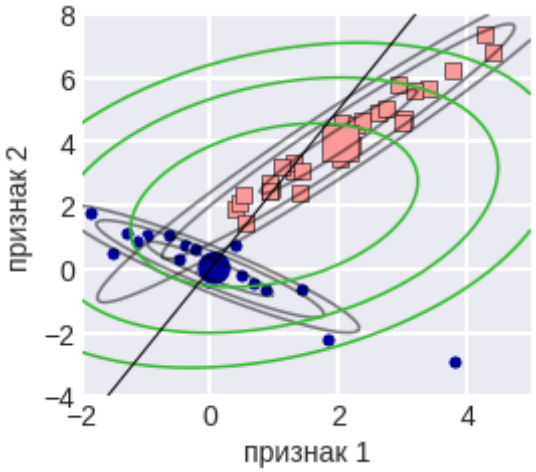


Рис. XB.20. Выборка и полученная прямая.

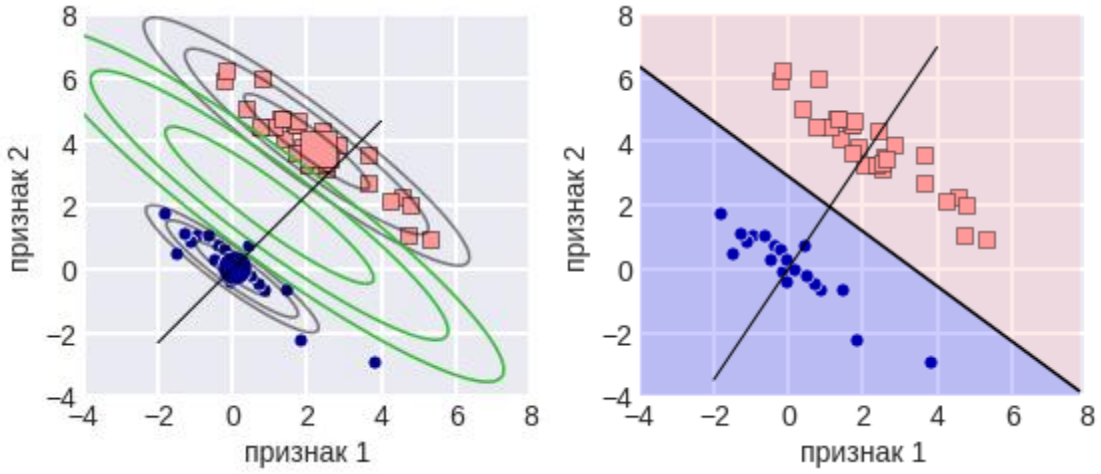


Рис. XB.21. Выборка и полученная прямая, а также полученная классификация.

На рис. XB.21 также показана разделяющая поверхность, которая соответствует гиперплоскости ортогональной разделяющей прямой, найденной линейным дискриминантом Фишера (LDA).

Приложения линейных классификаторов

1. Логистическая регрессия до сих пор используется в банковском скоринге, в задаче, где по описанию клиента нужно определить вероятность возврата кредита, хотя есть и более мощные методы (основанные на ансамблях решающих деревьев). Но здесь часто важна простота и интерпретируемость решения. Кроме того, подобные линейные методы позволяют получать совсем простые решения: т.н. **скоринговые карты**. Предположим, что, как показано на рис. XB.21, мы каждый признак превратили в категориальный. Например, вместо значения возраст у нас указание категории («до 30», «от 30 до 50», «старше 50»). Выполним для каждого категориального признака т.н. ONE-перекодировку: теперь вместо признака «возраст» у нас три бинарных признака: первый обращается в единицу только на клиентах с возрастом «до 30», второй – на клиентах с возрастом «от 30 до 50», третий – на клиентах с возрастом «старше 50»). Аналогично и для других признаков: вместо конкретного признака появляется несколько бинарных, их количество равно числу категорий для данного признака[1]. Теперь построим логистическую регрессию:

$$a(x) = 1 / (1 + \exp(-(w_0 + w_1 X_1 + \ldots + w_n X_n)).$$

Заметим, что результат монотонно зависит от линейной комбинации признаков и для принятия решения важна только она. Например, если мы выдаём кредит в случае, когда наша оценка вероятности превышает некоторый порог, то это эквивалентно тому, что порог должно превысить значение линейной комбинации признаков. Но каждый признак у нас бинарный характеристический. Таким образом, вес при этом признаке просто указывает вклад каждой категории в линейную комбинацию:

$$0 \cdot I[\text{возраст - до 30 лет}] + 35 \cdot I[\text{возраст - от 30 до 50 лет}] +$$

$$+ \ldots + 31 \cdot I[\text{трудовой стаж - более 10 лет}] \geq 100$$

На рис. XB.22 показана скоринговая карта – таблица, в которой указаны веса соответствующих признаков. Теперь принятие решения о выдачи кредита сводится к суммированию соответствующих весов и сравнению их с порогом. Например, к нам пришёл женатый клиент без образования, 29 лет, который

[1] На самом деле, это приведёт к линейно зависимым признакам, поэтому на практике одну категорию не кодируют, см. задачи.

работает 5 лет и никогда раньше не брал кредит. Мы вычисляем его скоринговый балл:

0 (возраст до 30) + 0 (образование не выше среднего) + 25 (в браке) + 0 (не брал раньше кредит) + 19 (стаж 5 лет) = 44.

Если мы даём кредит с 50 баллов, то можно сказать, что через год уже дадим (при прочих равных), т.к. трудовой стаж перевалит за 5 лет и возраст за 30.

Показатель	Значение показателя	Скоринг-балл
Возраст	До 30 лет От 30 до 50 лет Старше 50 лет	0 35 28
Образование	Среднее Среднее специальное Высшее	0 29 35
Состоит ли в браке	Да Нет	25 0
Брал ли кредит ранее	Да Нет	41 0
Трудовой стаж	Менее 1 года От 1 до 5 лет От 5 до 10 лет Более 10 лет	0 19 24 31

Рис. XB.22. Скоринговая карта [https://wiki.loginom.ru/articles/scorecard.html].

2. Многие линейные алгоритмы эффективно реализованы, также подходят для больших данных и данных в специальных форматах. Например, реализации sklearn.linear_model.SGDClassifier, sklearn.linear_model.SGDRegressor в библиотеке scikit-learn могут обучаться по разреженным (sparse) матрицам данным.

Вопросы и задачи

1. Алгоритм обучения персептрона получает решение за конечное число шагов, если объекты двух классов линейно разделимы. Что происходит, если они линейно не разделимы? Попробуйте написать детектор линейной неразделимости при обучении персептрона (перебирайте объекты не в случайном порядке, а по циклу).

2. Верно ли, что обучение линейного SVM эквивалентно нахождению кратчайшего отрезка, соединяющего выпуклые оболочки двух классов: по

такому отрезку выписывается уравнение разделяющей гиперплоскости и, наоборот, по уравнению находится отрезок?

3. Почему решение двойственной задачи в SVM существует, но в отличие от прямой, вообще говоря, неединственное? Приведите пример, когда есть несколько решений. Можно ли как-то выбрать из них лучшее?

4. Как можно модифицировать метод SVM, чтобы решение зависело не от граничных объектов, а от «типичных» объектов класса? Есть метод RVM, который устраняет указанный недостаток, но это не модификация, а отдельный метод.

5. Мы обосновали логистическую регрессию с помощью нормального распределения. Какие ещё распределения подойдут для обоснования? Можно ли избавиться от предположения равновероятности классов? Как меняется ответ при различных априорных вероятностях классов?

6. Обоснуйте применение функции softmax по аналогии с обоснованием сигмоиды.

7. Можно ли получить LDA из метода наименьших квадратов (который использовался у нас в линейной регрессии), выбрав целевые значения

$$\alpha \to \frac{m_{+1} + m_{-1}}{m_\alpha},$$

т.е. от задачи классификации мы переходим к задаче регрессии с такими метками?

8. Во всех линейных методах вектор параметров является линейной комбинацией признаковых описаний объектов. В методе SVM мы получили явную формулу, в которую входят описания опорных объектов. В остальных методах, если начинать с нулевого вектора параметров, то это следует из формул обновления весов в SGD. Какие плюсы и минусы такого представления? Можно ли улучшить сходимость методов обучения?

9. Есть мнение, что когда классы хорошо разделимы, оцениваемые параметры для логистической регрессии могу быть нестабильны, а линейный дискриминантный анализ (LDA) меньше подвержен этой проблеме. Попробуйте подтвердить или опровергнуть это. Также LDA рекомендуют использовать в признаковых пространствах малой размерности при

нормальном распределении классов. Также сделайте анализ такой рекомендации.

10. Полная ОНЕ-кодировка всех категориальных признаков приведёт к линейным зависимостям в признаках. С другой стороны, она позволяет получить скоринговую карту. Предложите способ построения скоринговой карты с помощью логистической регрессии, при котором нет линейно зависимых признаков.

Итоги

- Геометрический смысл линейных классификаторов – разделение точек гиперплоскостью.
- Есть стандартный подход в машинном обучении: использование суррогатной функции ошибки, которую проще минимизировать методами, основанными на градиентном спуске. Формулы для итерационного пересчёта весов во время обучения получаются простыми.
- В методе SVM строится, в некотором смысле, оптимальная разделяющая гиперплоскость. Но результат зависит от масштаба признаков и уровня шума в данных. Сама гиперплоскость зависит только от опорных объектов, которые находятся на границах класса.
- Формально для обучения и использования SVM не нужны признаковые описания объектов, нужно уметь вычислить скалярное произведение для любой пары объектов.
- Логистическая регрессия имеет вероятностное обоснование. Вид функции деформации выводится из формулы Байеса, а способ обучения из метода максимального правдоподобия.
- Линейный дискриминант Фишера генерирует признак (как линейную комбинацию исходных), на котором максимальный относительный разброс центров классов.

Код

```
X = list(np.array([[2,1], [-1, 0],
                   [-1, 1], [2, 2]]))
w = np.array([0, 0])
print ('w=', w)
change = True
while (change):
    change = False
    for x in X:
        if np.dot(x, w) <= 0:
            print ('w=', w, 'x=', x, '-')
            w += x
            change = True
print ('w=', w)
```

```
w=[ 0 0]
w=[ 0 0] x= [ 2 1] -
w=[ 2 1] x= [-1 0] -
w=[ 1 1] x= [-1 1] -
w=[ 0 2] x= [-1 0] -
w=[-1 2] x= [ 2 1] -
w=[ 1 3] x= [-1 0] -
w=[ 0 3] x= [-1 0] -
w=[-1 3]
```

Код. Персептроный алгоритм для решения системы линейных неравенств.

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X, y)
a = model.predict_proba(X_test)[:,1]
```

Код. Получение оценок вероятности принадлежности к классу 1.

```
from sklearn.discriminant_analysis import
LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis(solver="svd",
                                 store_covariance=True)
lda.fit(X, y)
```

Код. Применение LDA в scikit-learn.

Спасибо за внимание к книге!
Замечания по содержанию, замеченные ошибки и неточности можно написать в телеграм-чате
https://t.me/Dyakonovsbook